Report from the GI Dagstuhl Seminar 14433: Software Engineering for Self-Adaptive Systems

Thomas Vogel, Matthias Tichy, and Alessandra Gorla

CHALMERS | **UNIVERSITY OF GOTHENBURG**

Department of Computer Science and Engineering

Report from the GI Dagstuhl Seminar 14433: Software Engineering for Self-Adaptive Systems

Thomas Vogel, Matthias Tichy, Alessandra Gorla (editors)

Göteborg, Sweden 2014

Report from the GI Dagstuhl Seminar 14433: Software Engineering for Self-Adaptive Systems

Thomas Vogel[1], Matthias Tichy[2], and Alessandra Gorla[3] (editors)

[1] Hasso Plattner Institute, University of Potsdam, Germany
thomas.vogel@hpi.de

[2]Software Engineering Division, Chalmers | University of Gothenburg, Gothenburg, Sweden
matthias.tichy@cse.gu.se

[3]Saarland University, Germany
gorla@st.cs.uni-saarland.de

CHALMERS | UNIVERSITY OF GOTHENBURG

Report from the GI Dagstuhl Seminar 14433

Software Engineering for Self-Adaptive Systems

Edited by

Thomas Vogel[1], Matthias Tichy[2], and Alessandra Gorla[3]

1   **Hasso Plattner Institute, University of Potsdam, DE,**
    `thomas.vogel@hpi.de`
2   **Chalmers | University of Gothenburg, SE,**
    `matthias.tichy@cse.gu.se`
3   **Saarland University, DE,**
    `gorla@st.cs.uni-saarland.de`

---- **Abstract** --

Nowadays, software has become a key feature and driver for innovation of a wide range of products and services such as business applications, vehicles, or devices in various domains such as transportation, communication, energy, production, or health. Consequently, our daily lives highly depend on such software-intensive systems. This results in complex systems, which is even more stressed by integrating them to systems-of-systems or cyber-physical systems such as smart cities. Therefore, innovative ways of developing, deploying, maintaining, and evolving such software-intensive systems are required. In this direction, one promising stream of software engineering research is self-adaptation. Engineering self-adaptive systems is an open research challenge, particularly, for software engineering since it is usually software that controls the self-adaptation. This GI-Dagstuhl seminar focused on software engineering aspects of building self-adaptive systems cost-effectively and in a systematic and predictable manner. This includes typical software engineering disciplines such as requirements engineering, modeling, architecture, middleware, design, analysis, testing, validation, and verification as well as software evolution.

1   Executive Summary

Thomas Vogel
Matthias Tichy
Alessandra Gorla

Nowadays, software has become a key feature and driver for innovation of a wide range of products and services such as business applications, vehicles, or devices in various domains such as transportation, communication, energy, production, or health. Consequently, our daily lives highly depend on such software-intensive systems that therefore have to "become more versatile, flexible, resilient, dependable, energy-efficient, recoverable, customizable, configurable, and self-optimizing by adapting to changes that may occur in their operational contexts, environments and system requirements" [1, p. 1]. This results in complex systems,

which is even more stressed by integrating them to systems-of-systems or cyber-physical systems such as smart cities. Therefore, innovative ways of developing, deploying, maintaining, and evolving such software-intensive systems are required, which are major research challenges.

In this direction, one promising stream of software engineering research is self-adaptation, that is, systems that are able to automatically adjust their behavior and structure in response to changes in the environment or their own states and requirements. Engineering self-adaptive systems is an open research challenge, particularly, for software engineering since it is usually software that controls the self-adaptation [1].

Besides the notion of self-adaptive software systems, there are various other designations for such systems in the literature, such as self-healing, self-optimizing, self-managing, self-*, or autonomic systems. All of them share basic characteristics with respect to runtime adaptation though they oftentimes focus on specific concerns such as runtime failures (self-healing) or performance (self-optimization).

This GI-Dagstuhl seminar focused on software engineering aspects of building self-adaptive systems cost-effectively and in a systematic and predictable manner. This includes typical software engineering disciplines such as requirements engineering, modeling, architecture, middleware, design, analysis, testing, validation, and verification as well as software evolution (including software maintenance). Thus, the theme of the seminar jointly addressed the different software engineering disciplines to tackle the challenge of engineering self-adaptive systems.

The main goals of the seminar have been twofold. First, the seminar has brought together young researchers from the research community of "Software Engineering for Adaptive and Self-Managing System" (SEAMS)[1]. In the scope of this community, a series of Dagstuhl seminars on "Software Engineering for Self-Adaptive Systems"[2] have been organized while this seminar has aimed for young researchers to present their current research projects, to exchange experience and expertise, to discuss research challenges, and to develop ideas for future collaborations. Second, the proposed seminar has opened the SEAMS community to young researchers from related areas, particularly, the SPP 1183 "Organic Computing"[3], SFB 901 "On-The-Fly Computing"[4], SFB 1053 "MAKI – Multi-Mechanism Adaptation for the Future Internet"[5], and SPP 1593 "Design For Future – Managed Software Evolution"[6]. Thus, the seminar has fostered interaction and collaboration among young researchers working on self-adaptive software and related areas, which can be seen by the seminar's program.

The first two days of the seminar were spent on talks by the participants in session on "Requirements and Development", "Design and Development", "Control Theory and Synthesis", "Monitoring, Analysis and Decision-Making", "Verification, Validation, and Assurances", "Resource Efficiency and Performance", "Context-based Applications", and "Embedded and Cyber-Physical Systems". Based on these sessions, the participants formed break out groups that discussed hot topics in software engineering for self-adaptive systems during the last three days. These topics covered (1) the role of the user and context for self-adaptation, (2) distributed self-adaptive systems, (3) cyber-physical and self-adaptive

---

[1] Cf. SEAMS section on http://www.self-adaptive.org.
[2] Software Engineering for Self-Adaptive Systems: Assurances (2013): http://www.dagstuhl.de/13511
Software Engineering for Self-Adaptive Systems (2010): http://www.dagstuhl.de/10431
Software Engineering for Self-Adaptive Systems (2008): http://www.dagstuhl.de/08031
[3] http://www.organic-computing.de/SPP
[4] http://sfb901.uni-paderborn.de/
[5] http://www.maki.tu-darmstadt.de/sfb_maki/index.en.jsp
[6] http://www.dfg-spp1593.de/

systems, and (4) assurances for self-adaptive systems. Furthermore, some breakout groups formed subgroups to focus on specific research questions of the topics. Short reports from each (sub)group are listed in Section 2.

References

1   R. de Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, D. Weyns, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cukic, R. Desmarais, S. Dustdar, G. Engels, K. Geihs, K. Goeschka, A. Gorla, V. Grassi, P. Inverardi, G. Karsai, J. Kramer, A. Lopes, J. Magee, S. Malek, S. Mankovskii, R. Mirandola, J. Mylopoulos, O. Nierstrasz, M. Pezzè, C. Prehofer, W. Schäfer, R. Schlichting, D. B. Smith, J. P. Sousa, L. Tahvildari, K. Wong, and J. Wuttke. Software Engineering for Self-Adaptive Systems: A second Research Roadmap. In R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *LNCS*, pages 1–32. Springer, 2013.

Table of Contents

2 Breakout Groups

2.1 Smart Grid / Smart Home Self-adaptive Exemplar

Amel Belaggoun, Alexander Frömmgen, Alexander Schiendorfer, Matthias Tichy, and Sebastian Wätzoldt

Self-adaptive software (SAS) constitutes a key ingredient in the realization of cyber-physical systems that are inherently distributed. A very prominent example of such systems is found in the area of smart homes coupled with smart energy capabilities. Such a home can for instance offer production via photovoltaic plants or storage reserves in form of an electric vehicle. Designing efficient distributed control strategies is important - yet many approaches start from a very customized and hence incomparable initial setup. We therefore discussed an exemplar case study that could serve as a benchmark for evaluations of self-adaptations mechanisms. To be useful for SAS-developers, we included an initial meta-model and both a set of user stories that target requirements of quantities to be evaluated as well as a set of user stories highlighting possible collaboration scenarios (fully centralized vs. fully decentralized, proactive vs. reactive) inspired by ongoing research in the smart grid community.

2.2 Self-Adaptation in Highly Distributed Dynamic Systems

Ilias Gerostathopoulos, Sebastian Götz, Filip Krikava, Adnan Shahzada, and Romina Spalazzese

This breakout group focused on identifying the challenges of performing self-adaptation in highly distributed dynamic systems. This is a pressing issue in self-adaptive systems research, as proposed "smart" systems are increasingly built out of disparate entities (sensors and actuators) that feature a close connection to the physical world – so-called cyber-physical systems (CPSs). Examples are numerous: intelligent vehicle navigation, fleets of autonomous robots, emergency coordination systems, to mention just a few. CPSs are typically distributed at the physical space and feature no firm boundaries – they are open-ended. They are composed of loosely connected entities, that are often mobile. Grafting such systems with self-adaptive capabilities is a distinct challenge, which projects itself in all phases of the autonomic loop.

After debating on the different problems that can arise, the group focused on the issue of efficient information sharing between the entities of a self-adaptive CPS and how this affects the overall utility of the system. Our main assumption was that there is no central coordination point in our target CPS, but rather each entity follows its own decisions based on its partial view over the rest of the system. We also assumed that entities form ad-hoc collaboration groups to achieve common system-level objectives. Each partial view is then updated according to the information shared between the collaborating groups. In this frame, a separate autonomic loop is instantiated for each entity and consists of (i) monitoring the changes in the environment/nearby entities, (ii) creating a partial runtime model capturing the state of the world as viewed by the entity in question, (iii) deciding based on the partial runtime model and the adaptation logic, (iv) enacting the necessary changes to the runtime

behavior of the entity in question. In our discussions, we considered the case of a "smart crossroad", where each car decides whether to cross or wait at a crossroad according to its view over the positions and intentions of the other cars in the crossroad. In this case, the ad-hoc collaboration group between the cars that approach the crossroad has the objective to avoid collisions and increase the safety of the whole system.

Moving into the more concrete questions, the following challenges have been discussed in more detail:

1. **How to disseminate the information necessary to build the individual partial runtime models in an efficient way?** The problem is that communicating the information of each entity with every other entity in the CPS overloads the network and threatens the privacy of individual entities. Hence, there is a need for an efficient dissemination strategy for partial model building. Apart from the obvious yet naive strategy where every entity shares all its information with each and every one (mentioned above), we identified two more elaborate strategies: (i) each entity in a collaboration group shares all its information only with the group members; (ii) each entity in a collaboration group shares only the information that are relevant to its objectives and the objectives of the group only with the group members, i.e. shares only a fraction of its information.

2. **How to design runtime abstractions in order to support explicit information exchange between the entities of a collaboration group?** We have looked into the technical aspects for partial model sharing. Essentially, this is a problem of model querying with role-based access control and model merging. The models of the entity that represent its own view should allow to express what information are accessible to whom and what information are to be exchanged in each collaboration group. Once the information has been gathered, the next step is to find a way to merge it into the entity's existing partial view (at the receiving end). This becomes challenging when we consider that the same information can be provided by multiple entities and with different quality and precision. The merging has to therefore involve some trust schemes, temporal aspects as well as to support versioning and transactions.

2.3 Artifact-Centric Requirements Engineering for Self-Adaptive Systems

Alessia Knauss, Juan C. Muñoz-Fernández, Lorena Castañeda, Matthias Becker, Mahdi Derakhshanmanesh, Nina Taherimakhsousi, and Robert Heinrich

Given the dynamic nature of self-adaptive systems, traditional processes and artifacts for requirement engineering are not sufficient. Requirements engineering activities for self-adaptive systems have to take place at design time and runtime [2]. When developing a self-adaptive system, the designers have to define structural and behavioral adaptation points explicitly (where, when, what, how). In traditional approaches, a certain set of artifacts (e.g., requirements lists, goal models, feature models, statecharts, ...) is managed by requirements engineers. To achieve adaptivity, a self-adaptive system must be aware of this knowledge, i.e., in the form of runtime artifacts that are central constituents of the system and can be changed at runtime [1].

We envision that the interface between the design and runtime requirements engineering activities are artifacts that are created by the requirements engineer at design time and will be used and modified by the self-adaptive system at runtime. The goal of our breakout group was to gather a better and common understanding of (i) specific activities and (ii) related artifacts concerned with requirements engineering for self-adaptive systems. We especially focused on the roles of context and chiefly users context (users are one of the context) in the process of designing self-adaptation. Therefore, we specified a requirements engineering process that can be followed and a set of artifacts that can be used at design time and runtime. We constraint the process to seven steps:

1. Elicit requirements and goals.
2. Identify ambiguities in requirements and goals. Ambiguities might represent different influence factors.
3. Identify context that can have an influence on the system behaviour. Ambiguous requirements can be refined further through context-attributes (e.g., time and location).
4. Define context situations and variability in such situations.
5. Start with step 1 to identify more goals or step 2 to identify more ambiguities.
6. Define architectural significant requirements.
7. Operationalize the variability in the solution space through the definition of components.

Follow up: Along with a refined version of this initial process, we plan to give an overview of state-of-the-art and open issues in requirements engineering for self-adaptive systems.

We propose to leave the ambiguity in requirements on purpose and to implement the system without (fully) resolving the ambiguity. Usually, resolving the ambiguities is an essential part of the traditional requirements engineering activities. For the case of a self-adaptive system, however, we claim that making the definition of context-attributes that cause ambiguity explicit at design time (where possible) and letting the self-adaptive system resolve them at runtime will allow implementation of the system without fully hard coding, thereby leaving space for variability. This knowledge about potential adaptation points enables a self-adaptive system to adapt to emerging situations at runtime to deal with uncertainty.

As we inspect the sketched research challenge from different research areas such as requirements engineering, self-adaptive systems and modelling, we believe that other researchers in these communities will benefit from our work. Our work will give an overview of research gaps in this interdisciplinary topic of artifact-centric requirements engineering for self-adaptive systems.

References

1 N. Bencomo and J. W. et al. Requirements Reflection: Requirements as Runtime Entities. In *Int. Conf. on Software Engineering (ICSE'10)*, pages 199–202, 2010.

2 N. A. Qureshi, A. Perini, F. Bruno, K. Irst, N. A. Ernst, and J. Mylopoulos. Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems. In *Proceedings of 1st International Workshop on Requirements@RunTime*, pages 9–16, 2010.

2.4 Mitigating Data Leakage in Federated Self-Adaptive-Systems

Eric Schmieders, Christopher Bailey, and Iván Páez Anaya

Federated self-adaptive systems exhibit conflict in goals due to the existence of multiple stakeholders. As a result, these types of system are often vulnerable to exploitation and attacks. For instance, users of cloud providers have to fulfill privacy regulations that constrain the geographical locations of where personal data is processed or stored. In contrast to this, a cloud provider may dynamically transfer software components and data to alternative geographical locations to minimise cost. This is problematic as privacy reports show that data leakage rates are country specific. Consequently, sensitive data should be prevented from being stored or processed in countries with high data leakage to avoid privacy breaches. This is challenging in federated self-adaptive systems as third-party systems are not fully controlled by their users. To avoid this, we aim to tackle the problem of predicting and mitigating data migrations that imply high risks of data leakage. The prediction shall enable a self-adaptive system which relies on other self-adaptive systems to proactively mitigate future attacks. Specific problems comprise the monitoring and identification of malicious behaviour, especially unauthorized data access. The prediction model, build from past monitoring data, shall be utilized in order to predict future attacks and to execute mitigative actions proactively.

2.5 Assurances for Self-Adaptive Software Systems

Sinem Getir, Simos Gerasimou, Benedikt Eberhardinger, and Thomas Vogel

This breakout group focused on the aspect of *assurances* for self-adaptive software systems. We defined assurances as *"providing evidence that the self-adaptive software system fulfills its functional and non-functional requirements throughout its lifetime"*. The provision of *assurances* was classified by the software engineering community among the most important research objectives for self-adaptive systems [6] and was amid the research threads discussed extensively in the recent Dagstuhl seminars on "Software Engineering for Self-Adaptive Systems" [9, 8]. Our discussion was particularly driven to investigating the differences between assuring classical (i.e., static, non-adaptive) and self-adaptive software systems in order to work out differences, commons, and challenges.

Classical systems are characterized by *static* requirements, contexts, and software/system architectures that are fixed at design-time and do not change at runtime. In this regard, research in software engineering has focused for decades on providing techniques and tool-supported methodologies that, when applied collectively at design-time (i.e., before the system becomes operational), could assure compliance of the software system with its requirements [4]. Approaches within this category include, but are not limited to, model checking, testing, simulation, verification, safety analysis, theorem proving as well as inspection and development process techniques; see Nair et al. [12] for a taxonomy of these techniques. To this end, a software engineer can employ model-based simulation, using for example Queuing Networks [3], or can apply system and performance testing using Selenium and JMeter [13, 10], respectively.

Additionally, model checking and (probabilistic) verification can be useful in determining whether the system complies with its functional and/or non-functional requirements. Clearly, these techniques are applicable under the assumption that vital system characteristics, as for example system configuration, are known at design time and, more importantly, continue to hold at runtime.

The engineering of self-adaptive software systems brings to the forefront additional challenges due to the runtime dimension. The challenges we identified include continuously changing requirements and contexts in dynamic environments throughout the system's lifetime, and the inherent uncertainty of such changes, e.g., it is usually not feasible to anticipate *all* possible contexts, in which the system may operate, or to *precisely* characterize the current context of the system. Typically, to cope with ever-changing requirements, contexts, or system conditions, self-adaptive systems dynamically adjust their behavior often by reconfiguring their architecture at runtime [11]. That is, engineering decisions, for instance, concerning the configuration and architecture of the system are deferred to runtime [1, 2]. Consequently, there is imperative need to continue providing assurances that the system complies with its requirements at runtime [5], that is, while the system is providing service and adapting to changing situations (i.e., when the engineering decisions are made).

To investigate how well evidence techniques used in classical systems can cope with the challenges imposed by self-adaptive systems, we used as a running example the ZNN.com case study [7], a web-based client-server news system serving multimedia content to its customers. In the presented *rainbow architecture* [7] a controller is added to the server-side which realizes the self-adaptation. The investigation on comparing assurances built on the concrete performance requirement that *"after the server receives a request, the average response time should not exceed 2s per request"*.

We considered several existing evidence techniques and examined their applicability in assuring reliable operation in self-adaptive systems. Discussed techniques include testing, model checking, theorem proving and formal methods, as well as simulation and runtime verification. Clearly, there is a big overlap in the techniques used to provide assurances in classical and self-adaptive systems. For the latter type of systems, however, these techniques can be used if and only if critical system characteristics, for instance, requests arrival rate, are observable and measurable at runtime. Even if this is achievable, most of the existing evidence techniques have been engineered for design-time use and assuring dependable system operation at runtime is beyond their capacities.

Supporting high-integrity in self-adaptive systems entails revising existing evidence techniques as well as developing new approaches in order to tackle the emerging challenges of these systems. In fact, advanced techniques are needed, capable of handling uncertainty and incomplete knowledge at design-time as well as managing context changing and system evolution at runtime. Moreover, they need to be effective, efficient and scalable, with fast response times and low computation overheads. Possible methods to make these techniques appropriate for runtime use include compositional, incremental, and parametric approaches. Furthermore, they need to be capable of meeting the short time window given to carry-out the adaptation process. If this is infeasible, the techniques would ideally identify with high confidence the degree to which the system may invalidate its requirements.

We concluded that existing assurance techniques fall short of the challenges accompanying self-adaptive software systems. Hence, new evidence approaches and techniques are required for cooperatively providing assurances in this type of systems at design-time and runtime. We also identified the following key research question: *"are the techniques employed for providing assurances in self-adaptive systems a superset of the techniques used for the same*

purpose in classical systems?"

Following our initial findings, we plan to investigate in more detail the applicability of existing evidence techniques used in classical software systems for assuring dependable system operation in self-adaptive systems. We intend to carry out a survey and examine in depth the evidence techniques proposed in the literature and used in real self-adaptive systems, focusing our research in verification and testing. The expected outcome of our future work is to identify gaps in existing approaches as well as to discover connection points and overlapping areas between different evidence techniques, as for example, between testing and verification. We anticipate that a collection of assurance techniques should be used at design-time and made available at runtime, in order to achieve, with high confidence, dependable system operation in self-adaptive software systems.

References

1 J. Andersson, L. Baresi, N. Bencomo, R. de Lemos, A. Gorla, P. Inverardi, and T. Vogel. Software Engineering Processes for Self-Adaptive Systems. In R. de Lemos, H. Giese, H. Müller, and M. Shaw, editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *Lecture Notes in Computer Science (LNCS)*, pages 51–75. Springer, 2013.

2 L. Baresi and C. Ghezzi. The disappearing boundary between development-time and runtime. In *Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*, pages 17–22, New York, NY, USA, 2010. ACM.

3 S. Becker, L. Grunske, R. Mirandola, and S. Overhage. Performance Prediction of Component-Based Systems: A Survey from an Engineering Perspective. *Architecting Systems with Trustworthy Components*, pages 169–192, 2006.

4 B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, pages 592–605, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

5 R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive Software Needs Quantitative Verification at Runtime. *Commun. ACM*, 55(9):69–77, September 2012.

6 B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. Software engineering for self-adaptive systems. chapter Software Engineering for Self-Adaptive Systems: A Research Roadmap, pages 1–26. Springer-Verlag, Berlin, Heidelberg, 2009.

7 S.-W. Cheng. *Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 2008.

8 R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese. Software Engineering for Self-Adaptive Systems: Assurances (Dagstuhl Seminar 13511). *Dagstuhl Reports*, 3(12):67–96, 2014.

9 R. de Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, D. Weyns, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cukic, R. Desmarais, S. Dustdar, G. Engels, K. Geihs, K. Goeschka, A. Gorla, V. Grassi, P. Inverardi, G. Karsai, J. Kramer, A. Lopes, J. Magee, S. Malek, S. Mankovskii, R. Mirandola, J. Mylopoulos, O. Nierstrasz, M. Pezzè, C. Prehofer, W. Schäfer, R. Schlichting, D. B. Smith, J. P. Sousa, L. Tahvildari, K. Wong, and J. Wuttke. Software Engineering for Self-Adaptive Systems: A second Research Roadmap. In R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *LNCS*, pages 1–32. Springer, 2013.

10   H. Do, S. Elbaum, and G. Rothermel. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering*, 10(4):405–435, 2005.

11   J. Kramer and J. Magee. Self-managed systems: An architectural challenge. In *2007 Future of Software Engineering*, FOSE '07, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society.

12   S. Nair, J. L. de la Vara, M. Sabetzadeh, and L. Briand. An extended systematic literature review on provision of evidence for safety certification. *Information and Software Technology*, 56(7):689 – 717, 2014.

13   V. Stantchev. Performance evaluation of cloud computing offerings. In *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on*, pages 187–192, Oct 2009.

2.6 Topology Awareness for Self-Adaptive Systems

Antonio Filieri, Inti Gonzalez-Herrera, Alessandra Gorla, and Liliana Pasquale

Context awareness is a primary concern for self-adaptive systems. Such systems consider their execution context either to adapt their behavior to new (unexpected) situations, or to improve the satisfaction of existing requirements. Topology awareness concerns the description of the physical, digital, and interaction space a self-adaptive system operates in. It is getting momentum for applications requiring the system to reason about richer representations of the execution context, such as security and access control [10], location-dependent mobile applications [7, 11, 14], or web applications tailored to user profiles [6, 4]. Furthermore, additional non-functional properties, e.g., dependability and performance, can directly benefit of topology awareness.

Achieving topology awareness requires going through several challenges. First and foremost, automatic reasoning requires a formal characterization of context topology and relevant related concepts such as *containment*, *proximity*, and *reachability* [10]. Second, new verification and adaptation techniques should be developed in order to identify, mitigate and prevent violations of topology-related requirements. Finally, new monitoring infrastructures should be designed to update topology models at runtime.

We believe topology-aware adaptation techniques will have in the near future a significant impact on a number of emerging research and industrial areas, including Autonomous Robots [5, 13, 9, 15, 12], Cyber-Physical Systems [1], the Internet-of-Things [2, 3], and Cloud Computing [8, 16].

References

1   R. Al Ali, T. Bures, I. Gerostathopoulos, P. Hnetynka, J. Keznikl, M. Kit, and F. Plasil. Deeco: An ecosystem for cyber-physical systems. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 610–611, New York, NY, USA, 2014. ACM.

2   L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Comput. Netw.*, 54(15):2787–2805, Oct. 2010.

3 F. Bao and I.-R. Chen. Dynamic trust management for internet of things applications. In *Proceedings of the 2012 International Workshop on Self-aware Internet of Things*, Self-IoT '12, pages 1–6, New York, NY, USA, 2012. ACM.

4 A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Inf. Sci.*, 180(22):4290–4311, Nov. 2010.

5 M. Beetz, T. Schmitt, R. Hanek, S. Buck, F. Stulp, D. Schröter, and B. Radig. The agilo robot soccer team—experience-based learning and probabilistic reasoning in autonomous robot control. *Auton. Robots*, 17(1):55–77, July 2004.

6 G. Castellano, A. M. Fanelli, and M. A. Torsello. Computational intelligence techniques for web personalization. *Web Intelli. and Agent Sys.*, 6(3):253–272, Aug. 2008.

7 L. Jedrzejczyk, B. A. Price, A. K. Bandara, and B. Nuseibeh. On the impact of real-time feedback on users' behaviour in mobile location-sharing applications. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, pages 14:1–14:12, New York, NY, USA, 2010. ACM.

8 H. Kurra, Y. Al-Nashif, and S. Hariri. Resilient cloud data storage services. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 17:1–17:9, New York, NY, USA, 2013. ACM.

9 R. O'Grady, R. GroB, A. L. Christensen, and M. Dorigo. Self-assembly strategies in a group of autonomous mobile robots. *Auton. Robots*, 28(4):439–455, May 2010.

10 L. Pasquale, C. Ghezzi, C. Menghi, C. Tsigkanos, and B. Nuseibeh. Topology aware adaptive security. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS 2014, pages 43–48, New York, NY, USA, 2014. ACM.

11 S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications*, HotMobile '09, pages 3:1–3:6, New York, NY, USA, 2009. ACM.

12 A. V. Savkin and H. Teimoori. Decentralized navigation of groups of wheeled mobile robots with limited communication. *Trans. Rob.*, 26(6):1099–1104, Dec. 2010.

13 E. J. van Henten, J. Hemming, B. A. J. van Tuijl, J. G. Kornet, J. Meuleman, J. Bontsema, and E. A. van Os. An autonomous robot for harvesting cucumbers in greenhouses. *Auton. Robots*, 13(3):241–258, Nov. 2002.

14 U. Varshney. Location management for mobile commerce applications in wireless internet environment. *ACM Trans. Internet Technol.*, 3(3):236–255, Aug. 2003.

15 H. Yu and Y. Wang. Coordinated collective motion of groups of autonomous mobile robots with directed interconnected topology. *J. Intell. Robotics Syst.*, 53(1):87–98, Sept. 2008.

16 Y. Yu, J. Ni, M. H. Au, H. Liu, H. Wang, and C. Xu. Improved security of a dynamic remote data possession checking protocol for cloud storage. *Expert Syst. Appl.*, 41(17):7789–7796, Dec. 2014.

3.1 Integrating Predictive Analysis with Self-Adaptive Systems

Ivan Paez Anaya (University of Rennes 1, FR)

Recent literatures surveys have shown that the temporal aspects of self-adaptation is an area under explored. Furthermore, most of the current approaches are reactive. In some scenarios such as safety-critical systems (e.g. environmental monitoring), where the cost of failure could involve casualties, public and private property damage, the need for pro-activeness is obligatory. Current monitoring mechanisms in Self-adaptive systems are focus on instant values of the environment. This brings the problem of possible unnecessary adaptations that could happens in transients situations, than means that the event that triggers an adaptation is shorter than the time the system needs to carry on with the adaptation. Another scenario in which pure reactive adaptive adaptation is unfit is when we are facing seasonal behavior (e.g. hourly, daily, weekly, monthly phenomenons.). Failure to recognize these patterns can derive in oscillatory adaptations going from one state to another in a pendulous manner.

My research topic focus on the temporal aspects of self-adaptation aiming to improve the current state of self-adaptive systems by introducing predictive analytics techniques into the control loops. Techniques such as data mining, machine learning algorithms and predictors. The contributions of combining Self-Adaptive approach with Predictive Analytics are summarized in the three following points: 1) Multi objective selection of input variables for predictive models. 2) Balancing functional and nonfunctional properties of predictive models. 3) Predictive models as fitness functions in SBSE.

We integrate predictive analytics in three phases of the software process. An design time, we propose a multi-objective features input selection, with this, we are able to select those characteristics that best describe the behavior of the system. The machine learning techniques that can be used in this phase are data preprocessing, redimensioning and dimensional analysis. Detecting erroneous readings by removing out-layers form the data is also a beneficial in this phase. At runtime, we propose for the the self-adaptive systems classification and categorization techniques for drawing predictions that will feed the decision making mechanism in the control loop of the autonomic systems. As well as trade-off analysis of functional and non-functional properties of the system by using techniques such as correlation analysis of input variables.

3.2 Handling Insider Threats through Self-Adaptation

Christopher Bailey (University of Kent, GB)

Authorisation infrastructures are an integral part of any network where electronic resources require protection. As networks expand and organisations begin to federate access to their resources, authorisation infrastructures become increasingly challenging to manage. This talk explores the automatic adaptation of authorisation assets (such as, access control policies and subject access rights) in order to handle the identification and mitigation of insider

attacks, in regards to the abuse of privileged access. We demonstrate adaptation with an autonomic controller, capable of managing policy based role/attribute access control authorisation infrastructures. The autonomic controller implements a feedback loop to monitor the authorisation infrastructure in terms of its assets and subject behaviour, analyse a modelled system state for evidence of malicious behaviour, and act upon malicious behaviour by controlling future authorisation decisions. We discuss our evaluation techniques as well as preliminary results gained through the simulation of malicious behaviour within a deployed federated authorisation infrastructure.

3.3 Engineering Resource-Efficient and Elastic Self-Adaptive Systems

Matthias Becker (Universität Paderborn, DE)

Large business information systems nowadays run in cloud computing environments that have virtually unlimited computing resources. However, a resource-efficiency is still desired, because these computing resources are typically paid per use. Self-adaptive systems can autonomously adapt their architecture during operation and thus can potentially operate resource-efficient at all times. However, in current practice, self-adaptive systems are implemented and configured based on experience of software architects and rule-of-thumb. Design deficiencies limiting resource-efficiency and elasticity are often discovered in late development phases, i.e., in testing or even in operation.

We are developing SimuLizar, an engineering method that supports software architects to model resource-efficient and elastic self-adaptive multi-resource systems. The method supports software engineers with identifying and deciding quality trade-offs using the RELAX requirements language. Multiple relevant quality metrics can be predicted with SimuLizar such that design deficiencies in the software architecture and self-adaptation capabilities that limit resource-efficiency and elasticity can be identified already at design-time. Thus, unnecessarily high operation costs can be prevented and marginal costs for higher workloads become predictable.

3.4 Is it Useful to Make AUTOSAR Fully Dynamic?

Amel Belaggoun (CEA LIST, FR)

Increasingly, safety critical applications must function reliably and dynamically adapt their structure and /or behavior at runtime in response to changing conditions and environment. Runtime reconfiguration is a powerful mechanism to perform such adaptation. While few works have started developing techniques to perform software dynamic reconfiguration in real-time embedded systems and specifically in AUTOSAR, no work has provided an approach to perform safe runtime adaptation in AUTOSAR. I will present a first step in addressing this challenge.

3.5   Supporting Senior Shoppers with Self-Adaptive Personalized Web-Tasking

Lorena Castaneda (University of Victoria, CA)

E-commerce applications are highly popular among internet shoppers. However, changes on the variety of devices, services and applications add complexity during the web task execution. While technologically savvy people can easily handle this complexity, seniors might be challenged by numerous interaction steps. In particular, health conditions and degenerative diseases affect user's capability of interacting with devices or performing web tasks properly.

Personalized Web-Tasking (PWT) systems automate ordinary and repetitive web tasks while exploiting the personal context of the user to deliver personalized features. Current approaches of web automation and personalization rely on recorded information about users and their web interactions. In addition to this, self-adaptive PWT systems are capable to dynamically understand users' changing goals and situations, while adapting itself at runtime. Our research focuses on the context-aware and self-adaptive capabilities of PWT systems. The expected results of this work include: (1) a modelling specification for PWT, (2) a reference model for the design of PWT systems, and (3) a PWT model processing infrastructure.

This talk presents our recent contributions and ongoing work in the realization of self-adaptive PWT systems. We describe an e-commerce scenario focused on seniors in order to illustrate the importance of self-adaptive PWT systems capable to assist users in the fulfilment of personal goals using internet technologies.

3.6   The Vision of Model-Integrating Development

Mahdi Derakhshanmanesh (University of Koblenz-Landau, DE)

The development of complex software systems remains a challenge. Model-driven development approaches have been proposed and applied to achieve a certain degree of automation. Moreover, the use of models at runtime receives more and more attention as an enabling technology for separating concerns into explicit models and for controlled change operations on the running software, e.g., to realize self-adaptation. Despite their central role in the software development process, we observe that models are still not being treated as first-class entities: shipped software products still primarily consist of code. In order to leverage the full potential of models, we propose a realization concept for Model-Integrating Components (MoCos) that is compatible with existing component technologies. One can freely choose the portion of (executable) models and code when realizing a software component. These MoCos are the essential building blocks for a novel Model-Integrating Development (MID) approach to software engineering where models and code are artifacts with equal rights.

3.7 Testing Self-adaptive, Self-organising Systems

Benedikt Eberhardinger (Universität Augsburg, DE)

The characteristics of self-adaptive, self-organising systems lead to a significant higher
flexibility and robustness against a changing environment. Unfortunately, this flexibility
makes it hard to test these systems adequately. To assure their quality, however, it is
inevitable to do so.

Our approach for testing is based on the Corridor Enforcing Infrastructure (CEI), an
architectural pattern for SOAS. The CEI uses the concepts of feedback-loops to continuously
observe and control the system if necessary. On that account monitoring is used to achieve
situational awareness which is prerequisite to organise the system in a way to fulfil its
requirements. The bulk of concepts and techniques used in the CEI for controlling and
organising the system is triggered by the violation of constraints. An error-free CEI will
consequently guarantee a system that fulfils its requirements at every time. On this account,
we claim that testing SOAS can be achieved by testing the CEI. Here we benefit from specific
characteristics: we are able to reduce the relevant state space of the system to be tested
tremendously and to gain a clear distinction between correct and incorrect behaviour out of
the concepts of the CEI – essential for evaluating the tests. SOAS are highly distributed
systems, interacting locally to solve problems. This locality is exploited in our approach,
since many test cases can focus on single agents or small agent groups the system is composed
of; this focus makes it easier to execute and evaluate them. Furthermore, it is possible to
use the test results of local agents and agent groups to make a statement about the whole
system.

3.8 Automated control synthesis for dependable software adaptation

Antonio Filieri (Stuttgart University, DE)

Self-adaptation enables software to cope with changing and unpredictable environments.

Control theory provides a variety of mathematically grounded techniques for adapting the
behavior of dynamic systems. While it has been applied to specific software control problems,
it has proved difficult to define methodologies allowing non-experts to systematically apply
control techniques to create adaptive software.

However, at a convenient level of abstraction, broad classes of problems reveal similar
behavioral properties, allowing for the automated identification of a suitable dynamic model
and the synthesis of controllers with formally proved quality guarantees. As many practical
problems fall into treatable classes, automated control synthesis may draw in the near future
a new path toward dependable software adaptation.

3.9   Design Principles for Adaptive Communication Systems

Alexander Frömmgen (TU Darmstadt, DE)

Today's distributed systems have to work in changing environments and under different
working conditions. To provide high performance under these changing conditions, distributed
systems and communication systems have to implement adaptive behaviour.

My research concentrates on design principles for such adaptive communication systems.
Therefore, I am interested in the general description of adaptive behaviour, the possible
configurations and compositions of mechanisms, as well as the concrete execution of transi-
tions between these mechanisms. I am focusing on different network layers, especially the
application layer and network protocols.

My work is integrated in the Collaborative Research Centre "MAKI" (Multi-Mechanism
Adaptation for the Future Internet), which develops adaptive communication systems.

3.10   Runtime Quantitative Verification in Self-Adaptive AI Systems

Simos Gerasimou (University of York, GB)

Modern software systems need to self-adapt while providing service in order to cope with the
uncertain environment in which the systems operate, changing requirements and evolution
occurring in the system themselves. Consequently, trustworthy and dependable operation in
these systems is very important. This is more evident when these systems are deployed in
safety-critical or business-critical applications where they are expected to adhere to strict
performance, resource usage and other quality-of-service (QoS) requirements. Runtime
quantitative verification (RQV), a new approach for the analysis of systems exhibiting
stochastic behaviour, is a key technology in achieving both adaptation and continual compli-
ance with QoS requirements. Nevertheless, existing RQV techniques suffer from significant
overheads and cannot comply with the strict execution time and resource usage constraints
that characterise most runtime scenarios. The research we carry out aims to address some of
these challenges and also to extend the applicability of RQV to larger and more complex
self-adaptive software systems.

3.11   Adaptation in Ensemble-Based Component Systems: From
System Goals to Architecture Configurations

Ilias Gerostathopoulos (Charles University, CZ)

Ensemble-based component systems (EBCS) is a class of software component systems that
features the modelling constructs of autonomous components and ad-hoc groups (called
ensembles). These constructs have proven suitable for the development of networked, highly

dynamic cyber-physical systems, such as an intelligent decentralised car parking system or an emergency coordination system. In this talk, I will focus on the problem of designing such systems so that their situation-specific system-level goals are consistently mapped to implementation-level artefacts. I will present a design method – IRM-SA – that allows for such systematic design. The method establishes traceability between requirements and architecture in EBCS, while the outcome of the method (IRM-SA model) can be used at runtime to provide system adaptation in form of architecture reconfiguration.

3.12 Model-based Probabilistic Incremental Verification for Evolving Systems

Sinem Getir (University of Stuttgart, DE)

Probabilistic verification play an important role for verifying the quality attributes such as reliability, safety and performance. On problem however is, that system re-verification is needed whenever the software changes. Re-verifying of a changing model multiple times is expensive. Recently, incremental approaches have been found to be promising for the verification of evolving and and self-* systems. However, substantial improvements have not yet been achieved for evaluating structural changes in the model. This talk discusses an incremental verification framework that allows efficient evaluation of the changing models.

3.13 Multi-Quality Auto-Tuning: From Energy-neutrality to Robots and Roles

Sebastian Götz (TU Dresden, DE)

In this talk, I'll first outline a development and operation framework for self-optimizing software systems called MQuAT (Multi-Quality Auto-Tuning). Then, I'll focus on (software-)energy-optimization in particular, including a technique to save energy by frequency scaling and an approach to assess and compare the energy consumption of mobile applications. Additionally, I'll present a concept and prototype to realize energy-neutral software systems. Finally, I'll discuss work in the area of model-driven robot software engineering.

3.14 Resource reservation in pervasive middleware

Inti Gonzalez-Herrera (University of Rennes 1, FR)

The interest on distributed component-based software development is growing stronger due to, among other factors, the future Internet of Things (IoT). In a vision of the IoT, the infrastructure is shared among many stakeholders who deploy their own components

independently. As a result, no single stakeholder is able to centrally manage the computing resources and such a task must be carry on by the middleware. In a context where components share resources as memory and CPU cycles, the middleware must provide the each component with the resources it demands but it also must enforce quotas of consumption. To do so, the middleware monitors the system and executes adaptations as components' migration, modification of components' behaviour and modification of communication's patterns.

This research focuses in two aspects of the control loop, monitoring and execution of the plan. In particular, it aims at providing mechanisms for Java-based middleware to: i) perform resource consumption monitoring, ii) enforce resource reservation and, iii) execute runtime reconfiguration to provided distributed resource reservation. We describe mechanisms to perform scalable resource consumption monitoring and reservation. We also discuss approaches to identify the source of a faulty behaviour regarding resource usage. Finally, we present our vision on distributed decision making about resource reservation and how the paradigm of Models@runtime can be used as foundation for its implementation.

3.15 Self-Healing by Means of Intrinsic Redundancy

Alessandra Gorla (Universität des Saarlandes, DE)

Software can be redundant, in the sense that some operations are expected to behave like others, but their actual executions differ. This redundancy can be either deliberately introduced, as in the case of N-version programming, or intrinsically present due to common design and development practices. I present a technique that, by means of intrinsic redundancy, can make applications resilient to runtime failures, and thus achieve self-healing in software. The technique is intended to maintain a faulty application functional in the field while the developers work on permanent and radical fixes. It targets field failures in applications built on reusable components. In particular, the technique exploits the intrinsic redundancy of those components by identifying workarounds consisting of alternative uses of a faulty component that avoid the failure. The technique has been implemented and evaluated for Web [1] and Java [2] applications.

References

1 A. Carzaniga, A. Gorla, N. Perino, and M. Pezzè. Automatic workarounds for web applications. In *FSE'10: Proceedings of the 2010 Foundations of Software Engineering conference.* ACM, New York, NY, USA, 237–246.
2 A. Carzaniga, A. Gorla, N. Perino, A. Mattavelli, and M. Pezzè Automatic recovery from runtime failures. In *Proceedings of the 2013 International Conference on Software Engineering.* IEEE Press, 782–791.

3.16 Integrating Observation and Modeling Techniques to Support Adaptation and Evolution of Software-intensive Systems

Robert Heinrich (Karlsruhe Institute of Technology, DE)

The increased adoption of service-oriented technologies and cloud computing creates new challenges for the adaptation and evolution of long-living and software-intensive systems. Software services and cloud platforms are owned and maintained by independent parties. Software engineers and system operators of these systems only have limited visibility and control over the third-party elements. Traditional monitoring provides software engineers and system operators with execution observation data which are used as basis to detect anomalies. If the services and the cloud platform are not owned and controlled by the system engineers, monitoring the execution is a challenging task.

The aim of our research is to develop and validate advanced techniques which empower the system engineers to observe and detect upcoming quality flaws and anomalies of the execution of software-intensive systems. For this, we extend and integrate previous work on adaptive monitoring, software system modeling and analysis. We apply models@runtime as means to adjust the observation and anomaly detection techniques during system operation.

For demonstrating the feasibility and potential benefits gained and for providing feedback to guide the research, we continuously evaluated our results using the established research benchmark CoCoME.

3.17 Elicitation, Discovery and Evolution of Contextual Requirements

Alessia Knauss (University of Victoria, CA)

Increasingly complex socio-technical systems operate in continually changing environments. Such systems involve the interplay of human actors and technology. Changes in the context of the system might lead to changed user needs requiring a socio-technical system to adjust its behaviour. Due to uncertainty and increasingly changing operational environment such systems need to update their knowledge about end-user needs and context that influences the execution of end-user requirements. While self-adaptive systems are designed to address the challenges of changing context, the uncertainty in the operational environment poses outstanding challenges to capture and evolve requirements at runtime. By exploiting the concept of contextual requirements as composed of requirements and context as two entities that can evolve separately, my dissertation presents a framework for the analysis and discovery of contextual requirements at runtime. In three studies I explore the usefulness of existing techniques, i.e., requirements elicitation techniques and machine learning techniques, for the elicitation, discovery and evolution of contextual requirements. In my presentation I will give a short overview of my PhD thesis.

3.18 System-Level Abstractions for Integrating Control Mechanisms into Software Systems

Filip Krikava (University Lille 1 / LIFL, INRIA Lille, FR)

Control theory provides solid foundations and tools for designing and developing a reliable feedback control that drives software adaptations at runtime. However, the integration of the resulting control mechanisms is usually left to an extensive handcrafting of a non-trivial implementation code. This is a challenging task when considering the variety and complexity of contemporary distributed computing systems.

In our work, we aim to address this by providing flexible system-level abstractions in a form of runtime software models–i.e., models that are causally connected with running systems to monitor and manage specific aspects of their state and their environments. Such models allow to seamlessly observe and modify the underlying systems without the need for coping with low-level implementation and infrastructure details. Furthermore, techniques from model-driven engineering, such as model consistency checking and model transformations, can be used to systematically develop monitoring and reconfiguration parts of feedback control loops. As a result, this should allow researchers and engineers to experiment and to put easily in practice different self-adaptation mechanisms and policies.

3.19 Requirements Engineering Framework for Self Adaptive Software Systems

Juan Carlos Muñoz Fernández (Universidad Icesi, CO / Université Paris 1 Panthéon Sorbonne, FR)

Self-adaptation is a promising approach to manage the complexity of modern software systems that are required to adapt themselves autonomously. The dynamic nature of changing and evolving requirements is an important aspect in these systems. This kind of systems must be able to adapt to these changes, being it necessary to have a representation of their own requirements definition and to process their changes at runtime. Several results have shown that dynamic software product lines are feasible and useful in different software contexts including the modeling of requirements. From a literature review, we conclude that the modeling and characterization of requirements for SAS systems is an area that requires further exploration to be fully solved. The talk presents the current work the in the definition of the requirements engineering phase in the software development life cycle for self-adaptive systems based on the language proposed by Sawyer et al. [1]. The goal with an improved language and a framework is to provide a better the way of specifying requirements for SAS systems.

References

1   P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes. Using constraint programming to manage configurations in self-adaptive systems. *IEEE Computer*, 45(10):56–63, 2012.

3.20 Topology Aware Adaptive Security

Liliana Pasquale (Lero University, IE)

Adaptive security systems aim to protect valuable assets in the face of changes in their operational environment. They do so by monitoring and analysing this environment, and deploying security functions that satisfy some protection (security, privacy, or forensic) requirements. This talk suggests that a key characteristic for engineering adaptive security is the topology of the operational environment, which represents a physical and/or a digital space – including its structural relationships, such as containment, proximity, and reachability. For adaptive security, topology expresses a rich representation of context that can provide a system with both structural and semantic awareness of important contextual characteristics. These include the location of assets being protected or the proximity of potentially threatening agents that might harm them. Security-related actions, such as the physical movement of an actor from a room to another in a building, may be viewed as topological changes. The detection of a possible undesired topological change (such as an actor possessing a safe's key entering the room where the safe is located) may lead to the decision to deploy a particular security control to protect the relevant asset. By monitoring changes in topology at runtime one can identify new or changing threats and attacks, and deploy adequate security controls accordingly. The talk elaborates on the notion of topology and provides a vision and research agenda on its role for systematically engineering systems that satisfy their security, privacy and forensic requirements.

3.21 Constraints in Self-organizing, adaptive Systems

Alexander Schiendorfer (Universität Augsburg, DE)

Self-organizing, adaptive systems (SOAS) such as those considered in Organic Computing benefit from redundancies introduced to assert a certain fault tolerance and the ability to stay functional in case of failures. Resource allocation problems such as the distribution of a given predicted energy demand to a set of power plants are instances of tasks that can benefit from a self-organizing structure to deal with uncertainty and scalability. However, in order to control these systems (or even understand their behaviour), we need to devise algorithms and techniques dealing with their complex nature. Constraint programming and mathematical programming are successful paradigms designed to model a variety of satisfaction and optimization problems and solve them with generic algorithms. Therefore, a systematic integration of these declarative approaches into the engineering process of self-adaptive software is desirable. We present several techniques that are designed to incorporate preferences by means of soft constraints, physical relational models to add a semantic layer over pure constraint models and abstraction techniques to scale systems using hierarchies. An overview of this approach has been published in the Proceedings of the Second Organic Computing Doctoral Dissertation Colloquium 2014 and is available online.

3.22 Runtime Model based Privacy Checks of Cloud Services

Eric Schmieders (Paluno – University of Duisburg-Essen, DE)

Cloud elasticity enables cloud services to automatically adapt to workload changes by dynamically allocating and de-allocating hardware resources. Hardware allocation is achieved by virtual machine replication and migration that adds, removes, and re-deploys service components within or across data centers and thus alternates components and their deployments during runtime. Dynamically replicated or migrated components being part of complex cloud services may transfer data (even via intermediate components) to locations that are excluded by privacy policies. As dynamic adaptations are unpredictable during design time, changed compositions and deployments have to be observed and checked against privacy policies during runtime. In order to support the prevention of privacy violations, we focus on potential privacy violations that result from potential data transfers occurring once related cloud service functions are invoked. However, potential data flows are not observable, as they are not reflected in replication and migration events emitted by cloud infrastructures. This makes the detection of potential privacy violations problematic. To tackle this problem, we explore the utilization of runtime models in order to reflect cloud services, their components, and interactions, which allows for reasoning on potential privacy violations among the cloud service composition. Based on the key idea of detecting privacy violations on runtime models, this talk will discuss two main technical challenges faced when putting the approach into practice and, further, points out how we address these challenges: (a) the challenge of observing cloud elasticity despite the limited visibility of cloud internals and (b) the challenge of checking realistic cloud services of large size and complexity.

3.23 A Comprehensive Framework for the Development of Dynamic Smart Spaces

Adnan Shahzada (Politecnico di Milano, IT)

The conception of efficient smart spaces requires a reliable framework for their design, implementation, testing, and deployment. The development life cycle of these smart spaces differs greatly from the conventional software systems, as it poses various challenges during each development phase such as: provision of appropriate design abstractions, integration and coordination of heterogeneous components, incremental evaluation and deployment of the system, and formation of ad-hoc network to cater dynamism in diverse smart spaces. There have been numerous solutions proposed to solve different aspects related to smart spaces, but we still lack a conceptual framework that provides tools for the whole development life cycle. This work presents a generic and comprehensive development framework that allows the developer to move seamlessly from a fully virtual, simulated solution to a completely deployed system. The framework uses a group-based organization for the coordination and cooperation among the heterogeneous components of a smart space. It also provides interfaces to surrogate certain sets of system components through external simulators, and ease the deployment of physical elements. The usefulness of the proposed framework has

been demonstrated with the help of diversified smart spaces.

3.24 Automated approaches to build self-adaptive systems

Romina Spalazzese (Malmö University, SE)

Nowadays, our living environment is pervaded by a wide variety of heterogeneous digital systems that are connected to the Internet. The number and kind of connected systems have been always increasing and this growth trend will continue in the future. In this context, systems meet and know each other dynamically, when they want to start to interoperate to achieve some goal. Key challenges, thus, are to enable systems to interoperate seamlessly and to guarantee some properties despite context changes. Given the huge heterogeneity and dynamism characterizing the described environment, automated solutions appear to be the only way to face such challenges timely and with the needed level of flexibility.

In this talk I will present our recent and current work and future research directions. I will describe a solution for the automated synthesis of connectors (a) that takes into account performance concerns during the synthesis process and (b) whose synthesized connectors are (self-)adaptive with respect to runtime performance requirement changes. Moreover, I will talk about our ongoing work on an approach to automatically build context-aware (self-)adaptive systems. Our approach elicits automatically relevant context-variability and properly extends the systems to make them self-adaptive to context changes.

3.25 Context-based Face Recognition for Smart Application

Nina Taherimakhsousi (University of Victoria, CA)

Face recognition is a challenging computational task which assigns an identity to detected faces. There has been prodigious improvement on face recognition in recent years. At first, researchers focused on face recognition under controlled conditions. Facilitated by classic datasets, researchers investigated face recognitions invariant to changes in pose, facial expression and illumination. Despite their initial success, substantial face recognition research is required in less-controlled or uncontrolled conditions, such as in personal photo collections, web images and videos, as found in on-line social networks. Searching over a large set of images amplifies the need more robust face recognition methods.

I propose to investigate real-time context-based approaches for face recognition available to improve face recognition processes. My expected contributions include a context-based face recognition framework for mobile devices to improve accuracy rate. For example, with the help of location sensors on the mobile devices annotated images with location information. I propose an algorithm that exploit context to reduce the search space of face recognition and, therefore, achieve better result. Photos are clustered by locations on the server that are then associated with a face classifier. A client can send a query to the server by uploading a subject image with the location information. The face recognition algorithm suitable

for narrow the search space on the server. My results are quite promising with respect to accuracy and queries are answered in real-time.

3.26   Dependability Improvement by Self-Adaptation

Matthias Tichy (Chalmers | University of Gothenburg, SE)

Cyber-physical systems often have to satisfy high dependability requirements, e.g., reliability and safety. Self-adaptation enables a system to sense the current state related to dependability (e.g., failures of system parts), analyze the state, plan corrective actions and execute them. In this talk, we present approaches addressing safety and availability. First, with respect to safety, we show how a system after detection of a failure executes architectural reconfigurations to avoid a hazard. We present an analysis approach to check whether the reconfiguration is fast enough to avoid that the hazard happens. Second, we combine graph transformations and the planning domain definition language in an approach to compute repair plans to improve the availability. We show evaluation results of how many systems a central planning system can concurrently supply with repair plans.

3.27   Model-Driven Engineering of Self-Adaptive Software with EUREMA

Thomas Vogel (Universität Potsdam, DE)

The development of self-adaptive software requires the engineering of an adaptation engine that controls the underlying adaptable software by feedback loops. The engine often describes the adaptation by runtime models representing the adaptable software and by activities such as analysis and planning that use these models. To systematically address the interplay between runtime models and adaptation activities, runtime megamodels have been proposed. A runtime megamodel is a specific model capturing runtime models and adaptation activities. In this article, we go one step further and present an executable modeling language for ExecUtable RuntimE MegAmodels (EUREMA) that eases the development of adaptation engines by following a model-driven engineering approach. We provide a domain-specific modeling language and a runtime interpreter for adaptation engines, in particular feedback loops. Megamodels are kept alive at runtime and by interpreting them, they are directly executed to run feedback loops. Additionally, they can be dynamically adjusted to adapt feedback loops. Thus, EUREMA supports development by making feedback loops explicit at a higher level of abstraction and it enables solutions where multiple feedback loops interact or operate on top of each other and self-adaptation co-exists with offline adaptation for evolution.

3.28 Modeling Collaboration in Cyber-Physical Systems

Sebastian Wätzoldt (Universität Potsdam, DE)

Cyber-Physical Systems (CPS) evolved from embedded systems that combine physical processes with computing. Whereas embedded systems are mostly closed, self-contained and do not expose the computing capability to the outside, an increasing number of devices and demands of combined functionalities led to a more and more interconnection of embedded systems to so-called networked embedded systems. As a consequence, such isolated control systems and afterwards interconnected embedded systems become open to its environment and build different variants of cyber-physical systems that integrate the cyber (software) and physical part. While a CPS is a federation of open, ubiquitous systems that dynamically and automatically reconfigure themselves and cooperate with other CPS, the modeling of this cooperation (collaboration) has yet not gained the necessary attention as the modeling of feedback loops traditionally emphasized in embedded system design still dominates the thinking. By lifting collaborations to the level of first class entities when modeling CPS and by covering diverse forms of feedback loop interactions such as hierarchy as well as adaptation a modeling language can enable the appropriately exploring of the solution space for CPS. In this talk, I will present ideas of my current research concerning the modeling of collaboration in CPS.

4 Participants

- Ivan Paez Anaya
University of Rennes 1, FR
- Christopher Bailey
University of Kent, GB
- Matthias Becker
Universität Paderborn, DE
- Amel Belaggoun
CEA LIST, FR
- Lorena Castaneda
University of Victoria, CA
- Mahdi Derakhshanmanesh
University of Koblenz-Landau,
DE
- Benedikt Eberhardinger
Universität Augsburg, DE
- Antonio Filieri
Stuttgart University, DE
- Alexander Frömmgen
TU Darmstadt, DE
- Simos Gerasimou
University of York, GB

- Ilias Gerostathopoulos
Charles University, CZ
- Sinem Getir
University of Stuttgart, DE
- Sebastian Götz
TU Dresden, DE
- Inti Gonzalez-Herrera
University of Rennes 1, FR
- Alessandra Gorla
Universität des Saarlandes, DE
- Robert Heinrich
Karlsruhe Institute of
Technology, DE
- Alessia Knauss
University of Victoria, CA
- Filip Krikava
University Lille 1 / LIFL, INRIA
Lille, FR
- Juan Carlos Muñoz Fernández
Universidad Icesi, CO /
Université Paris 1 Panthéon
Sorbonne, FR

- Liliana Pasquale
Lero University, IE
- Alexander Schiendorfer
Universität Augsburg, DE
- Eric Schmieders
Paluno – University of
Duisburg-Essen, DE
- Adnan Shahzada
Politecnico di Milano, IT
- Romina Spalazzese
Malmö University, SE
- Nina Taherimakhsousi
University of Victoria, CA
- Matthias Tichy
Chalmers University of
Technology, SE / University of
Gothenburg, SE
- Thomas Vogel
Universität Potsdam, DE
- Sebastian Wätzoldt
Universität Potsdam, DE

Photo by Schloss Dagstuhl

Acknowledgements